IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re: Application of:	:	
Mark Allen Grubbs	:	
	: Before the Examiner:	
Serial No: 10/621,951	: Charles Edward Lu	
	:	
Filed: 07/17/2003	: Group Art Unit: 2163	
	:	
Title: PERFORMANCE-ENHANCING	: Confirmation No.: 1546	
SYSTEM AND METHOD OF	:	
ACCESSING FILE SYSTEM	:	
OBJECTS	:	

<u>APPELLANTS' BRIEF UNDER 37 C.F.R. 41.37</u>

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

     In a Final Office Action dated July 21, 2006, the Examiner rejected the claims in this Application under 35 U.S.C. §103(a) as being unpatentable over Sinha in view of Pinkoski and further in view of Achiwa et al. On December 22, 2006, Applicants appealed this decision. In an Office Action dated April 13, 2007, the Examiner reopened the Application's prosecution. In that Office Action, the Examiner rejected the claims under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha and further in view of Applicants' admitted prior art. Applicants/Appellants do not believe that the Duchamp, Kleiman, Sinha and Applicants' admitted prior art, alone or in combination, teach the claimed invention and request reinstatement of the Appeal.

     This Appeal Brief is submitted pursuant to a Notice of Appeal filed on July 13, 2007 in accordance with 37 CFR §41.31 and no fee is required.

AUS920030463US1

BRIEF FOR APPLICANTS - APPELLANTS

(i)

Real Party in Interest

The real party in interest is International Business Machines Corporation (IBM), the assignee.

(ii)

Related Appeals and Interferences

There are no other appeals or interferences known to appellants, appellants' representative or assignee, which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(iii)

Status of Claims

Claims 1 - 20 are finally rejected and Claims 1 – 20 are being appealed.

(iv)

Status of Amendment

An "Amendment after Final" was not filed.

(v)

Summary of Claimed Subject Matter

The invention, as claimed in Claim 1, provides a method of providing a performance-enhancing way of accessing frequently-accessed file system objects (see page 1, lines 9 – 11, page 5, lines 3 and 4). The method comprises the steps of determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system (see page 11, lines 19 – 31, page 12, lines 9 – 24, page 15, line 24 to page 12, line 7);

AUS920030463US1

entering the pathname of the at least one file system object into a memory system; and cross-referencing the pathname of the at least one file system object in the memory system with its inode number thereby enabling the inode number to be obtained with one memory access (see page 11, lines 19 – 31 and items 1405 and 1410 in Fig. 14).

The invention, as claimed in Claim 8, provides a computer program on a computer readable medium for enhancing performance of a system when frequently-accessed file system objects are being accessed. The computer program comprises code means for determining at least one frequently-accessed file system object in a file system upon the file system being mounted at a mount point on the system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system (see page 11, lines 19 – 31, page 12, lines 9 – 24, page 15, line 24 to page 12, line 7); code means for entering the pathname of the at least one file system object into a memory system; and code means for cross-referencing the pathname of the at least one file system object in the memory system with its i-node number thereby allowing the inode number to be obtained with one memory access (see page 11, lines 19 – 31 and items 1405 and 1410 in Fig. 14). The code means plus functions are the steps in Figs. 15 and on page 15, line 21 to page 16, line 7

The invention, as claimed in Claim 15, provides a system. The system comprises at least one storage system (main memory 1904, disk 1926, tape 1928, CD-ROM 1930 or memory 1924) for storing code data; and at least one processor (processor 1902 in Fig. 19) for processing the code data to determine at least one frequently-accessed file system object in a file system upon the file system being mounted at a mount point on the system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system (see page 11, lines 19 – 31, page 12, lines 9 – 24, page 15, line 24 to page 12, line 7), to enter the pathname of the at least one file system object into a memory system, and to cross-reference the

AUS920030463US1

pathname of the at least one file system object in the memory system with its inode number thereby allowing the inode number to be obtained with one memory access (see page 11, lines 19 – 31 and items 1405 and 1410 in Fig. 14).

<div align="center">(vi)</div>

<div align="center">Grounds of Rejection to be Reviewed on Appeal</div>

Whether it was proper to reject Claims 1, 4, 5, 7, 8, 11, 12, 14, 15, 18 and 19 under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha and further in view of Applicants' admitted prior art

Whether it was proper to reject Claims 2, 3, 9, 10, 16 and 17 under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha, further in view of Applicants' admitted prior art and further in view of Nevarez

Whether it was proper to reject Claims 6, 13 and 20 under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha, further in view of Applicants' admitted prior art and further in view of Stern

<div align="center">(vii)</div>

<div align="center">Arguments</div>

It is a well settled law that in considering a Section §103 rejection, the subject matter of the claim "as a whole" must be considered and analyzed. In the analysis, it is necessary that the scope and contents of the prior art and differences between the art and the claimed invention be determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966).

Whether it was proper to reject Claims 1, 4, 5, 7, 8, 11, 12, 14, 15, 18 and 19 under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha and further in view of Applicants' admitted prior art

Claims 1, 8 and 15

Duchamp purports to teach an optimistic lookup of whole network file system (NFS) paths in a single operation. According to Duchamp, virtual file system (VFS) lookup algorithm requires that every component in a pathname be checked and analyzed when resolving a pathname. The component-by-component analysis of the path causes a large number of lookups to go to the server over wire and thus degrades performance. To counter this performance degradation, Duchamp adds a path-to-vnode cache at the VFS level and augments NFS to lookup whole paths whenever the path cache misses by adding a path-lookup call to the NFS protocol. By adding the path-lookup, the MOUNT protocol is altered such that, at mount time, the file server indicates if it can handle path-lookup, and if so, which types of pathname syntax it understands (see the last paragraph of section 2.1). The client then stores this information for that mount.

Accordingly, at mount time Duchamp is only interested in knowing whether or not the file server can handle path-lookup and not whether or not there is a file system object (frequently-accessed or otherwise) in the file system. Further, Duchamp teaches the step of cross-referencing pathnames to vnodes and not to inodes.

Therefore, Duchamp does not teach, show or suggest the step of **determining at least one _frequently-accessed file system object_ in a file system upon mounting the file system at a mount point on a computer system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system** and **cross-referencing the pathname of the at least one file system object in**

the memory system with its inode number thereby enabling the inode number to be obtained with one memory access as in the claimed invention.

The Examiner admitted that Duchamp does not teach inode numbers and frequently accessed files but stated that Kleiman is cited to show that a "vnode" is well-known in the art as reading on an inode since Kleiman defines a vnode as a file system-independent inode, that Appellants admitted that an inode is identified by a unique number called an inode number and that Sinha teaches allowing a user to specify a frequently accessed object. Therefore, the Examiner reasoned, combining the teachings of Duchamp, Kleiman, Appellants' admitted prior art and Sinha shows the claimed invention. Appellants respectfully disagree.

Firstly, in accordance with the teachings of Kleiman, a UNIX file system has an inode layer which is separated into a dependent inode and an independent inode. The independent inode part is renamed a vnode by Kleiman. Thus, a vnode, as used by Kleiman, is much less than an inode since it encompasses only the independent part of the inode and not both the dependent and the independent parts of the inode. Therefore, a "vnode" does not read on an inode.

Secondly, Sinha purports to teach a pathname resolution method for providing fixed speed of file accessing in computer network. According to the purported teachings of Sinha, each user of a node of a distributed system can selectively specify one of a plurality of modes of path name resolution for use in accessing a specific file from a node of the system. The selected mode provides a fixed, minimum access time for the file and utilizes a name cache within the main memory of that node. In one of the selected modes, the name cache contains pathnames of frequently accessed files which are cross-referenced to the location of the files. For example, if the file is on a remote system, the location of the file will contain an identification of the remote system and information to locate the file on that remote system.

Since Duchamp states that at mount time it is determined whether or not the file server can handle path-lookup, Kleiman discloses a vnode while Sinha teaches that name caches contain pathnames of frequently accessed files which are cross-referenced to the location of the files, there is no reason for one skilled in the art to combine the teachings of Sinha, Kleiman with those of Duchamp absent a specific teaching or suggestion to do so in the references.

After all in *In re Fritch*, 972 F.2d 1260, 23 USPQ 2d 1780, 1783–84 (Fed. Cir. 1992), the Court ruled that "[o]bviousness cannot be established by combining the teachings of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. Under section 103, teachings of references can be combined *only* if there is some suggestion or incentive to do so." (quoting *ACS Hosp. Systems, Inc. v. Montefiore Hosp.*, 732 F.2d 1572, 1577, 221 USPQ 929, 933 (Fed. Cir. 1984)). . . . The mere fact that the prior art may be modified in the manner suggested by the Examiner does not make the modification obvious unless the prior art suggested the desirability of the modification.

In this case, there is not any teaching or suggestion in any of the references to combine their teachings. Hence, Appellants submit that the Examiner has impermissibly combined the teachings of the references.

Notwithstanding the fact that there is not any reason for anyone skilled in the art to combine the teachings of the references, if one were to combine the teachings of Sinha with those of Duchamp, Kleiman and Appellants' admitted prior art, the resulting combination would teach entering frequently-accessed files in the path-to-vnode cache of Duchamp. It would not, however, teach the step of ***determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system*** as in the claimed invention.

Therefore, Applicants submit that Claims 1, 8 and 15 are patentable over the combination of the teachings of Duchamp, Kleiman, Sinha and Applicants' admitted prior art.

Whether it was proper to reject Claims 2, 3, 9, 10, 16 and 17 under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha, further in view of Applicants' admitted prior art and further in view of Nevarez

<u>Claims 2, 9 and 16</u>

Claims 2, 9 and 16 recite the limitations that the determining step includes the step of obtaining from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

Nevarez purports to teach a method for storing a database in extended attributes of a file system. Accordingly, a database containing management, maintenance, control, and other user-defined data that are associated with a directory entry of the file system is created. The database is stored inside the extended attribute or sets of extended attributes. Each extended attribute includes a header for defining the extended attribute that comprises a first plurality of fields. It also includes a plurality of records for storing data. In turn, each record of the plurality of records comprises a second plurality of fields. The extended attribute is identified by a string stored in a directory entry of a directory entry table for the file system. The string comprises a predetermined substring denoting the extended attribute having the database architecture of the present invention. The first plurality of fields of the extended attribute includes fields for storing a length of the header, a version number of the header, a revision number of the header, the number of extended attributes in the database, and the number of records in the plurality of records in the database. The second plurality of fields includes fields for storing a name of a record, data, a record

AUS920030463US1

length, a length of the record name, and a length of the data. Both the name and data are stored as strings.

However, Nevarez does not teach the step of ***obtaining from an extended attribute file a list of pathnames <u>to be entered into the memory system</u>, the extended attribute file being associated with the mounted file system***.

Consequently, combining the teachings of Nevarez with those of Duchamp, kleiman, Sinha and Applicants' admitted prior art does not teach the invention as claimed in Claims 2, 9 and 16.

Nonetheless, it should be pointed out that Sinha specifically teaches that a user makes the determination as to whether a file will be frequently accessed. When the user makes the determination, the user supplies the pathname of the file to the system in order for the name of the file to be entered into the name cache (see col. 7, lines 41 – 54). By combining the teachings of Nevarez with those of kleiman, Sinha and Applicants' admitted prior art to show the invention as claimed in Claims 2, 9 and 16, the Examiner is asserting that the determining step includes the step of obtaining from an extended attribute file a list of pathnames to be entered into the memory system wherein the extended attribute file is associated with the mounted file system.

Applicants submit that if the user supplies the pathname of the file, there is no reason for the name of the file to be obtained from an extended attribute file. In other words, the teachings of Nevarez cannot be combined with those of Sinha, Pinkowski and Achiwa et al. to show the claimed invention and the Examiner has impermissibly combined the teachings of the references.

Whether it was proper to reject Claims 6, 13 and 20 under 35 U.S.C. §103(a) as being unpatentable over Duchamp as evidenced by Kleiman in view of Sinha, further in view of Applicants' admitted prior art and further in view of Stern

Claims 6, 13 and 20

Claims 6, 13 and 20 contain the limitations that the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

The Examiner stated that Stern teaches these limitations. Appellants disagree.

Stern teaches removing any cached entries for files on a file system that is umounted from the DNLC. However, the entries that are in the DNLC are entries of edges (i.e., each element of a pathname).

Thus, if a pathname has three elements (e.g. /usr/lib/libc.a), there will be entries in the DNLC for each edge (i.e., each edge usr, lib and libc.a will have an entry in which the edge name is cross-referenced to its inode number in the DLNC cache). It is these entries that will be removed. But note that, if the file system lib/libc.a was mounted on root file /user, when the lib/libc.a is unmounted only entries for lib and libc.a will be removed from the DNLC cache.

By contrast, only one entry will be removed from the MENC cache whether or not lib/libc.a was mounted on a root file. The one entry will be the /user/lib/libc.a which is cross-referenced to the inode number of the libc.a file. This is quite different from the teachings of Stern.
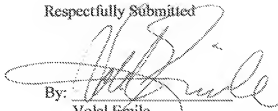
Thus, Appellants submit that there would not be any reason for anyone skilled in the art to combine the teachings of Duchamp, Kleiman, Sinha, Applicants' admitted prior with those of Stern since Stern does not teach the claimed invention as in Claims 6, 13 and 20.

Hence, Claims 6, 13 and 20 are patentable over the teachings of Duchamp as evidenced by Kleiman in view of Sinha, further in view of Applicants' admitted prior art and further in view of Stern.

Based on the foregoings, Applicants request passage to issue of all the claims in the Application.

AUS920030463US1

Respectfully Submitted

By: _____

Volel Emile
Attorney for Applicants
Registration No. 39,969
(512) 306-7969

(viii)

Claims Appendix

1. (Previously presented) A method of providing a performance-enhancing way of accessing frequently-accessed file system objects comprising the steps of:

    determining at least one frequently-accessed file system object in a file system upon mounting the file system at a mount point on a computer system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system;

    entering the pathname of the at least one file system object into a memory system; and

    cross-referencing the pathname of the at least one file system object in the memory system with its inode number thereby enabling the inode number to be obtained with one memory access.

2. (Previously presented) The method of Claim 3 wherein the pathnames in the extended attribute file are relative to the mount point.

3. (Previously presented) The method of Claim 1 wherein the determining step includes the step of obtaining from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

4. (Previously presented) The method of Claim 1 wherein the determining step includes the step of obtaining the pathname from a user.

5.   (Previously presented) The method of Claim 1 wherein the determining step includes the step of monitoring accesses to a file system object within a certain time span.

6.   (Previously presented) The method of Claim 1 wherein the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

7.   (Previously presented) The performance-enhancing method of Claim 1 wherein a pathname of a file system object and its cross-referenced inode number is removed from the memory system when a user so ordered.

8.   (Previously presented) A computer program on a computer readable medium for enhancing performance of a system when frequently-accessed file system objects are being accessed comprising:

code means for determining at least one frequently-accessed file system object in a file system upon the file system being mounted at a mount point on the system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system;

code means for entering the pathname of the at least one file system object into a memory system; and

code means for cross-referencing the pathname of the at least one file system object in the memory system with its i-node number thereby allowing the inode number to be obtained with one memory access.

AUS920030463US1

9.   (Previously presented) The computer program of Claim 10 wherein the pathnames in the extended attribute file are relative to the mount point.

10.   (Original) The computer program of Claim 8 wherein the determining code means includes code means for obtaining from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

11.   (Previously presented) The computer program of Claim 8 wherein the determining code means includes code means for obtaining obtaining the pathname from a user.

12.   (Original) The computer program of Claim 8 wherein the determining code means includes code means for monitoring accesses to a file system object within a certain time span.

13.   (Previously presented) The computer program of Claim 8 wherein the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

14.   (Original) The computer program of Claim 8 wherein a pathname of a file system object and its cross-referenced inode number is removed from the memory system when a user so ordered.

15.   (Previously presented) A system comprising:

at least one storage system for storing code data; and

AUS920030463US1

at least one processor for processing the code data to determine at least one frequently-accessed file system object in a file system upon the file system being mounted at a mount point on the system, each file system object having a pathname and an inode number, the inode number for locating the file system object on a storage system, to enter the pathname of the at least one file system object into a memory system, and to cross-reference the pathname of the at least one file system object in the memory system with its inode number thereby allowing the inode number to be obtained with one memory access.

16. (Previously presented) The system of Claim 17 wherein the pathnames in the extended attribute file are relative to the mount point.

17. (Original) The system of Claim 15 wherein the code data is further processed to obtain from an extended attribute file a list of pathnames to be entered into the memory system, the extended attribute file being associated with the mounted file system.

18. (Previously presented) The system of Claim 15 wherein the code data is further processed to obtain the pathname from a user.

19. (Original) The system of Claim 15 wherein the code data is further processed to monitor accesses to a file system object within a certain time span.

20. (Previously presented) The system of Claim 15 wherein the pathname of the file system object and its cross-referenced inode number are removed from the memory system when the file system containing the file system object is unmounted.

AUS920030463US1

(ix)

<u>Evidence Appendix</u>

None.

(x)

Related Proceedings Appendix

None.